



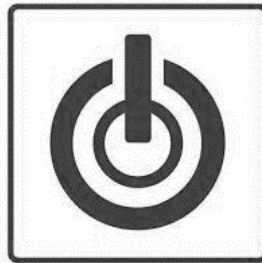
UNIVERSIDAD NACIONAL  
AUTÓNOMA DE MÉXICO



FACULTAD DE ESTUDIOS  
SUPERIORES ARAGÓN

Ingeniería en Computación

COMPILADORES



Tarea 7 Mini interprete de C

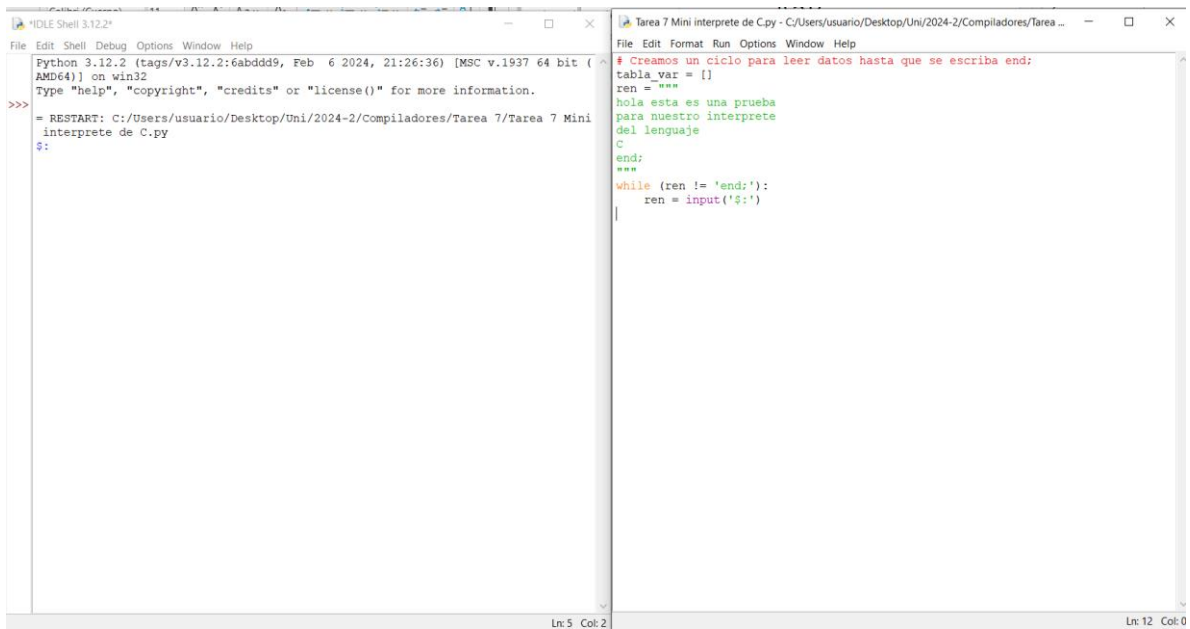
Profesor: Marcelo Pérez Medel

Leonardo Olvera Martínez

Grupo: 2608

Fecha: jueves 18 de abril de 2024

Ejecutamos la primera parte del código, como podemos observar, el código nos indica que ingresemos



```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/usuario/Desktop/Uni/2024-2/Compiladores/Tarea 7/Tarea 7 Mini interprete de C.py
$:
```

```
# Creamos un ciclo para leer datos hasta que se escriba end;
tabla_var = []
ren = ""
hola esta es una prueba
para nuestro interprete
del lenguaje
C
end;
"""
while (ren != 'end;'):
    ren = input('$:')
|
```

A continuación explicamos detalladamente el código para crear el interprete del lenguaje C.

```
Tarea 7 Mini interprete de C.py - C:/Users/usuario/Desktop/Uni/2024-2/Compiladores/Tarea ...
File Edit Format Run Options Window Help

# Creamos una clase variable para asignar nuevos objetos de tipo variable
class Variable:
    def __init__(self, nombre, tipo):
        self.nombre = nombre
        self.tipo = tipo
        self.valor = None

# Esta funcion sirve para agregar los objetos de tipo variable a un array
def agrega_var(tabla_var, nombre, tipo):
    tabla_var.append(Variable(nombre, tipo))
    pass

# Funcion para verificar si existe ya existe una variable con el mismo nombre
def existe_var(tabla_var, nombre):
    # Creamos una bandera
    encontrado = False
    for v in tabla_var:
        # Buscamos el nombre de la variable en nuestro array
        if v.nombre == nombre:
            # Si la encontramos cambiamos la bandera a True
            encontrado = True
    return encontrado

# Funcion para reasignar un valor a una variable ya existente
def set_var(tabla_var, nombre, valor):
    # Buscamos si existe la variable
    if existe_var(tabla_var, nombre):
        for v in tabla_var:
            # Buscamos la variable con el nombre que queremos reasignar
            if v.nombre == nombre:
                # Cambiamos el valor
                v.valor = valor
    # Si no existe la variable
    else:
        # Retornamos un error
        print('variable ', nombre, 'no encontrada')
        return None
```

```
# Funcion para reasignar un valor a una variable ya existente
def set_var(tabla_var, nombre, valor):
    # Buscamos si existe la variable
    if existe_var(tabla_var, nombre):
        for v in tabla_var:
            # Buscamos la variable con el nombre que queremos reasignar
            if v.nombre == nombre:
                # Cambiamos el valor
                v.valor = valor
    # Si no existe la variable
else:
    # Retornamos un error
    print('variable ', nombre, 'no encontrada')
    return None

# Funcion para imprimir todas las variables almacenadas
def imprime_tabla_var(tabla_var):
    print()
    print('Tabla de variables')
    print('nombre\t\ttipo\t\tvalor')
    for v in tabla_var:
        # Imprime los campos de los objetos
        print(v.nombre, '\t\t', v.tipo, '\t\t', v.valor)
    return None
```

Por problemas con el IDL, la tarea se continuó en Visual Studio.

A continuación, se muestra el código explicado agregando el código para que pueda aceptar print y valores en las variables.

```
Tarea 7 Mini interprete de C.py X
C: > Users > usuario > Desktop > Uni > 2024-2 > Compiladores > Tarea 7 > Tarea 7 Mini interprete de C.py > ...
1  # Creamos una clase variable para asignar nuevos objetos de tipo variable
2  # Asignamos el valor del atributo valor para que se haga referencia al del objeto
3  class Variable:
4      def __init__(self, nombre, tipo, valor):
5          self.nombre = nombre
6          self.tipo = tipo
7          self.valor = valor
8
9  # Esta funcion sirve para agregar los objetos de tipo variable a un array
10 # Agregamos el atributo VALOR
11 def agrega_var(tabla_var, nombre, tipo, valor):
12     tabla_var.append(Variable(nombre, tipo, valor))
13     pass
14
15 # Funcion para verificar si existe ya existe una variable con el mismo nombre
16 def existe_var(tabla_var, nombre):
17     # Creamos una bandera
18     encontrado = False
19     for v in tabla_var:
20         # Buscamos el nombre de la variable en nuestro array
21         if v.nombre == nombre:
22             # Si la encontramos cambiamos la bandera a True
23             encontrado = True
24     return encontrado
25
26 # Funcion para reasignar un valor a una variable ya existente
27 def set_var(tabla_var, nombre, valor):
28     # Buscamos si existe la variable
29     if existe_var(tabla_var, nombre):
30         for v in tabla_var:
31             # Buscamos la variable con el nombre que queremos reasignar
32             if v.nombre == nombre:
33                 # Cambiamos el valor
34                 v.valor = valor
35     # Si no existe la variable
36     else:
37         # Retornamos un error
38         print(f'Variable {nombre} no existe')
```

```
27 def set_var(tabla_var, nombre, valor):
```

```

36     else:
37         # Retornamos un error
38         print('variable ', nombre, 'no encontrada')
39         return None
40
41 # Funcion para imprimir todas las variables almacenadas
42 def imprime_tabla_var(tabla_var):
43     print()
44     print('    Tabla de variables')
45     print('nombre\t\ttipo\t\tvalor')
46     for v in tabla_var:
47         # Imprime los campos de los objetos
48         print(v.nombre, '\t\t', v.tipo, '\t\t', v.valor)
49     return None
50
51 #                                     SEPARA TOKENS
52
53 # Funcion para detectar si es un caracter especial
54 def es_simbolo_esp(caracter):
55     return caracter in "+-*,;,:!#=%&/({}[ ]<>=<==""
56
57 # Funcion para ver si es un separador
58 def es_separador(caracter):
59     return caracter in " \n\t"
60
61 # Funcion para detectar si se trata de un id
62 def es_id(cad):
63     return (cad[0] in "_abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")
64
65 # Funcion para detectar si se trata de una palabra reservada
66 def es_pal_res(cad):
67     palres = ["int","real","string", 'print', 'read', 'tabla']
68     return (cad in palres)
69
70 # Funcion para ver si se trata de un tipo de dato
71 def es_tipo(cad):

```

```
71 def es_tipo(cad):
72     tipos = ["int", "real", "string"]
73     return (cad in tipos)
74
75 def esEntero(dato):
76     if isinstance(dato, int):
77         return True
78     else:
79         return False
80
81 # Funcion para separar en tokens nuestro codigo
82 def separa_tokens(linea):
83     # Revisamos que el codigo que nos dieron no sea menor a 3 caracteres
84     if len(linea) < 3:
85         return []
86     else: # Si es mayor a 3 caracteres
87         tokens = []
88         tokens2 = []
89         dentro = False
90         # Revisamos caracter por caracter
91         for l in linea:
92             # Si es simbolo especial
93             if es_simbolo_esp(l) and not(dentro):
94                 # agregalo a la lista de tokens
95                 tokens.append(l)
96             # si es simbolo especial o separador
97             if (es_simbolo_esp(l) or es_separador(l)) and dentro:
98                 tokens.append(cad)
99                 dentro = False
100             # Si es simbolo especial
101             if es_simbolo_esp(l):
102                 # Agregalo a la lista
103                 tokens.append(l)
104             # Si no es simbolo especial y no es separador asumimos que es un id
105             if not (es_simbolo_esp(l)) and not (es_separador(l)) and not(dentro):
106                 #cambiamos la bandera para guardar el id
107                 dentro = True
```

```
82 def separa_tokens(linea):
106     #cambiamos la bandera para guardar el id
107     dentro = True
108     cad=""
109     # Si no es simbolo especial y no es separador y la bandera esta activada
110     if not (es_simbolo_esp(1)) and not (es_separador(1)) and dentro:
111         # Almacenamos el valor en una variable
112         cad = cad + l
113
114     # Creamos una bandera
115     compuesto = False
116     # Recorremos nuestros tokens menos 1
117     for c in range(len(tokens)-1):
118         if compuesto:
119             compuesto = False
120             continue
121         # Si en el token que leemos encontramos un operador y el siguiente es un =
122         if tokens[c] in "<>!" and tokens[c+1]=="=":
123             # agregamos a nuestro arreglo de tokens el token c mas =
124             tokens2.append(tokens[c]+"=")
125             compuesto = True
126         # Si no, continuamos sin agregar el =
127         else:
128             tokens2.append(tokens[c])
129     # Agrega el último token a la lista tokens2
130     tokens2.append(tokens[-1])
131
132     # Recorre la lista tokens2 para encontrar los decimales
133     for c in range(1,len(tokens2)-1):
134         # si el token es . busca si el anterior y consecuente es entero
135         if tokens2[c]=="." and esEntero(tokens2[c-1]) and esEntero(tokens2[c+1]):
136             # Si es asi, elimina el anterior y consecuente y los une con c
137             tokens2[c]=tokens2[c-1]+tokens2[c]+tokens2[c+1]
138             # Marcamos los tokens para borrarlos
139             tokens2[c-1]="borrar"
140             tokens2[c+1]="borrar"
141     # Borramos los anteriores y consecuentes
```



Tarea 7 Mini interprete de C.py X

C: > Users > usuario > Desktop > Uni > 2024-2 > Compiladores > Tarea 7 >  Tarea 7 Mini interprete de C.py >  separa\_tok

```

82 def separa_tokens(linea):
140     tokens2[c+1]="borrar"
141     # Borramos los anteriores y consecuentes
142     porBorrar = tokens2.count("borrar")
143     for c in range(porBorrar):
144         tokens2.remove("borrar")
145     # Limpiamos el array tokens
146     tokens=[]
147     # Creamos una bandera
148     dentroCad = False
149     # Creamos una cadena vacia
150     cadena = ""
151     # Recorremos tokens2
152     for t in tokens2:
153         # Si la bandera esta activa
154         if dentroCad:
155             # Revisamos si termina en comillas
156             if t[-1]=="'":
157                 # Agrega el token a la cadena actual
158                 cadena=cadena+" "+t
159                 # Agrega la cadena actual, sin las comillas, a la lista de tokens
160                 tokens.append(cadena[1:-1])
161                 # Cambia bandera a False
162                 dentroCad = False
163             else:
164                 # Agregar token a la cadena
165                 cadena = cadena+" "+t
166         # Si el token actual comienza con "
167         elif ((t[0]=="'")):
168             # Guardamos el valor t en la cadena
169             cadena= t
170             # Actuamos la bandera
171             dentroCad = True
172         # Si no es una cadena
173         else:
174             tokens.append(t)
175     return tokens

```

## Tarea 7 Mini interprete de C.py X

C: &gt; Users &gt; usuario &gt; Desktop &gt; Uni &gt; 2024-2 &gt; Compiladores &gt; Tarea 7 &gt; Tarea 7 Mini interprete de C.py &gt; ...

```
82 def separa_tokens(linea):
174     tokens.append(t)
175     return tokens
176
177     # Creamos un array
178     tabla_var = []
179     # Creamos una cadena vacia
180     ren = ""
181     # Mientras la cadena no sea igual a end; el codigo continua
182     while (ren != 'end;'):
183         # Solicitamos el ingreso de una línea de código
184         ren = input('$:')
185         # Separamos lo que ingresaron en tokens
186         tokens = separa_tokens(ren)
187         # Verificamos si el primer token es un ID
188         if es_id(tokens[0]):
189             # Verificamos si el primer token es una palabra reservada
190             if es_pal_res(tokens[0]):
191                 # Verificamos si el primer token es un tipo
192                 if es_tipo(tokens[0]):
193                     # Verificamos si el segundo token es un ID
194                     if es_id(tokens[1]):
195                         # CODIGO PARA AGREGAR VALORES A LA TABLA
196                         if(tokens[2] == '=' and tokens[4] == ';'):
197                             agrega_var(tabla_var, tokens[1], tokens[0], tokens[3])
198                         else:
199                             # Agregamos a nuestra tabla el nombre de nuestro variable y valor
200                             agrega_var(tabla_var, tokens[1], tokens[0], None)
201             # Si el token es read
202             elif tokens[0] == 'read':
203                 # Si el siguiente token es un ( y el siguiente un Id y el siguiente )
204                 if tokens[1] == '(' and es_id(tokens[2]) and tokens[3] == ')':
205                     # Leer un valor del usuario e insertarlo en la variable correspondiente
206                     leido = input()
207                     set_var(tabla_var, tokens[2], leido)
208             # Si el token es read
209             elif tokens[0] == 'tabla':
```

## Tarea 7 Mini interprete de C.py X

C: &gt; Users &gt; usuario &gt; Desktop &gt; Uni &gt; 2024-2 &gt; Compiladores &gt; Tarea 7 &gt; Tarea 7 Mini interprete de C.py &gt; ...

```
190     if es_pal_res(tokens[0]):
191         # Verificamos si el primer token es un tipo
192         if es_tipo(tokens[0]):
193             # Verificamos si el segundo token es un ID
194             if es_id(tokens[1]):
195                 # CODIGO PARA AGREGAR VALORES A LA TABLA
196                 if (tokens[2] == '=' and tokens[4] == ';'):
197                     agrega_var(tabla_var, tokens[1], tokens[0], tokens[3])
198                 else:
199                     # Agregamos a nuestra tabla el nombre de nuestro variable y valor
200                     agrega_var(tabla_var, tokens[1], tokens[0], None)
201             # Si el token es read
202             elif tokens[0] == 'read':
203                 # Si el siguiente token es un ( y el siguiente un Id y el siguiente )
204                 if tokens[1] == '(' and es_id(tokens[2]) and tokens[3] == ')':
205                     # Leer un valor del usuario e insertarlo en la variable correspondiente
206                     leído = input()
207                     set_var(tabla_var, tokens[2], leído)
208             # Si el token es read
209             elif tokens[0] == 'tabla':
210                 # Imprimir la tabla de variables con los valores
211                 imprime_tabla_var(tabla_var)
212
213             # CODIGO PARA IMPRIMIR USANDO print();
214             # Verificar si el token es la palabra reservada 'print'
215             elif tokens[0] == 'print':
216                 if tokens[1] == '(' and es_id(tokens[2]) and tokens[3] == ')' and tokens[4] == ';':
217                     # Verificamos si la variable existe antes de imprimir
218                     if existe_var(tabla_var, tokens[2]):
219                         for v in tabla_var:
220                             # Busca la variable
221                             if v.nombre == tokens[2]:
222                                 print(v.valor)
223                     else:
224                         # Mostrar un mensaje de error si la variable no se encuentra
225                         print(['Error Variable', tokens[2], 'no encontrada'])
```